VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

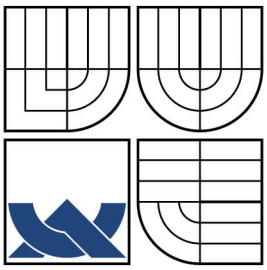
JAVAME APLIKACE PRO SÁZENÍ NA SPORTOVNÍ ZÁPASY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

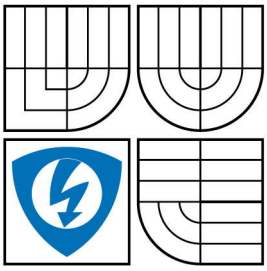
AUTOR PRÁCE
AUTHOR

FRANTIŠEK KUBALÍK

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

JAVA ME APLIKACE PRO SÁZENÍ NA SPORTOVNÍ ZÁPASY

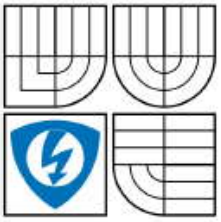
JAVA ME APPLICATION FOR BETTING ON SPORT RESULTS

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — FRANTIŠEK KUBALÍK
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR ČÍKA
SUPERVISOR

BRNO 2009



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	František Kubalík	***ID:***	72950
Ročník:	3	***Akademický rok:***	2008/2009

NÁZEV TÉMATU:

JavaME aplikace pro sázení na sportovní zápasy

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s možnostmi on-line sázení na sportovní události. Navrhněte aplikaci pro mobilní terminály, která bude uživateli umožňovat sázení na aktuální sportovní zápasy. Daný návrh realizujte v jazyku JavaME. Funkčnost aplikace bude závislá na centrální databázi, kterou taktéž vytvoříte. Bude obsahovat přihlašovací údaje jednotlivých klientů, výši kreditu a kurzy na plánované sportovní zápasy.

DOPORUČENÁ LITERATURA:

[1] Yuan, J., Y.: Enterprise J2ME: Developing Mobile Java Applications, Indiana, Prentice Hall PTR 2003, ISBN 978-0131405301
[2] Wells, M., J.: Java ME Game Programming, 2E, Florence, Course Technology PTR 2007, ISBN 978-1598633894

Termín zadání:	9.2.2009	***Termín odevzdání:***	2.6.2009

Vedoucí práce: Ing. Petr Číka

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

Tato bakalářská práce se věnuje návrhu a realizaci aplikace pro mobilní terminály, která umožňuje uživateli sázet na sportovní zápasy. Funkčnost této aplikace je závislá na centrální databázi, jejíž návrh je také součástí této práce. V první části jsou popsány základní vlastnosti programovacích jazyků, které byly pro realizaci projektu použity. Postupně je tedy popsán jazyk Java, SQL a PHP. Zmíněna je také správná tvorba tabulek a popis problémů, které mohou při návrhu databáze nastat. Další kapitola pak popisuje samotnou realizaci celého projektu. Je zde popsána zvolená metoda a podrobnější popis realizace jednotlivých částí, ze kterých se projekt skládá. Tato část obsahuje také testování vytvořené aplikace.

KLÍČOVÁ SLOVA

J2ME, programování, midlet, Java, databáze, SQL, PHP, skript, MIDP, sázení

ABSTRACT

This bachelor thesis deals with the design and implementation of applications for mobile terminals, which allows user to bet on sports games. The functionality of this application is dependent on a central database, whose proposal is also part of this work. The first section describes the basic features of programming languages, which were used for the implementation of the project. Gradually, it is described in Java, SQL and PHP. The correct formation of tables and a description of problems that may arise in the design of the database are mentioned. Next chapter describes the actual implementation of the project. It described the methods and a more detailed description of the implementation of the various parts the project comprises. This section also contains a test created by the application.

KEYWORDS

J2ME, programming, midlet, Java, database, SQL, PHP, script, MIDP, betting

KUBALÍK, F. JavaME *aplikace pro sázení na sportovní zápasy*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2009. 35 s. Vedoucí bakalářské práce Ing. Petr Číka.

Prohlášení autora o původnosti práce

Prohlašuji, že svou bakalářskou práci na téma „Java ME aplikace pro sázení na sportovní zápasy“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… …… ……

podpis autora

Poděkování

Děkuji vedoucímu bakalářské práce Ing. Petru Číkovi, za velmi užitečnou metodickou pomoc a cenné rady při zpracování práce.

V Brně dne …… ……

podpis autora

OBSAH

Úvod

Mobilní telefony jsou v dnešní době pro většinu lidí určitě nepostradatelnou součástí každodenního života. Mnozí z nich však nemají ani tušení, čím vším musí mobilní telefony (lépe řečeno jejich programové vybavení) projít, aby pomocí nich mohli provádět tak samozřejmé operace, jako jsou například posílání textových zpráv, nebo spouštění různých aplikací, sloužících k čemukoliv. I když je mnoho prostředků, které vývoj těchto aplikací usnadňují, je nutné ošetřit všechny žádoucí a hlavně nežádoucí stavy, ke kterým může v běhu aplikace dojít, což s sebou nese mnoho zkoušení a voleb těch správných metod. Návrh jedné z těchto aplikací je hlavním předmětem této bakalářské práce.

Celá bakalářská práce je rozdělena na několik částí. Před samotným návrhem řešení daného problému a tím i splnění cílů závěrečné práce, je potřeba se seznámit se základními pojmy, s nimiž se můžeme v průběhu řešení setkat. Také je potřeba si ukázat důležitá teoretická fakta, jejichž vlastnosti budeme muset uplatnit. To je úkolem první části, nazvané Základní pojmy a teorie. Jejím cílem však není popsat všechny informace týkající se daného tématu, pouze zeširoka se s danými pojmy seznámit. Hlavním jádrem celého projektu bude použití programovacího jazyka Java a také komunikace s databází, vytvořenou v programovacím jazyce SQL (Structured Query Language). Celou tuto komunikaci bude mít na starosti skript, který bude naprogramován v jazyce PHP (Personal Home Page, později Hypertext Preprocessor) a právě na něm bude záviset funkčnost celé aplikace. Tato část bude tedy zaměřena právě na pojmy, které s těmito jazyky souvisí a to od jejich struktury, až po důležité vlastnosti a nástroje, které nám jejich použití usnadní. Ve druhé části pak bude popsáno, co vše bylo k realizaci projektu potřebné provést a jak vše bylo řešeno, dojde zde také k podrobnému testování samotné aplikace. Následující poslední část pak bude obsahovat zhodnocení celého projektu.

1 ZÁKLADNÍ POJMY A TEORIE

1.1 Programovací jazyk Java

1.1.1 Vznik

Na počátku devadesátých let společnost Sun Microsystems vyvinula nový programovací jazyk Oak. Jeho přednosti byly vysoká spolehlivost a ne příliš přehnané nároky na paměť a výkonnost procesoru, což se logicky uplatnilo zejména u malých zařízení. I když tento jazyk vlastně vycházel z jazyka C++, odstraněním některých jeho rysů (například ukazatele a správa paměti) se snížila náchylnost k programátorským chybám a tím se výrazně zvýšila spolehlivost [1], [2].

Očekávaná poptávka společností Sun po tomto druhů zařízení se ovšem nekonala. Časem byl jazyk Oak přejmenován na jazyk Java a protože v té době začal do podvědomí lidí vstupovat internet, z původního záměru použití Javy se přešlo na vývoj aplikací právě tímto směrem. Z licenčních důvodů došlo později ještě k jednomu přejmenování, a to na Java 2 [1], [2].

1.1.2 Zpracování programu

Způsob zpracování programu v Javě probíhá v pěti fázích. Jsou to editování, překlad (kompilace), zavedení (load), ověřování (verifikace) a provádění, takže vzniká další rozdíl oproti ostatním programovacím jazykům, kde k ověřování nedochází. Tím je zvýšena bezpečnost spuštěného programu.

Java platforma se skládá ze dvou hlavních částí. První je Java Virtual Machine (JVM), neboli virtuální stroj Javy, který se skládá z runtime systému a interpreteru. Runtime systém realizuje vazbu na hardware a k interpreteru se dostaneme za okamžik. Druhá část Java platformy je Application Programming Interface (API), neboli aplikační programové rozhraní, obsahující knihovny pro psaní programů.

Překlad neprobíhá do strojového jazyka počítače, ale do pseudojazyka nazývaného byte-code. Tento jazyk je nezávislý na cílovém počítači, takže je úplně jedno, na jakém počítači vytvořený program poběží. To s sebou nese důležitou vlastnost, tou je přenositelnost.

Přeložený program, respektive byte-code, je uložen v souboru s vyhrazenou příponou .class. Tento soubor je pak z disku zaveden do paměti a současně probíhá ověření byte-code. Po ověření je program spouštěn pomocí interpreteru. Problémem interpretovaných jazyků je jejich pomalost ve srovnání s kompilovanými jazyky, což Java částečně řeší pomocí tzv. JIT (Just In Time) kompilátorů, které v době zavádění programu z disku do paměti počítače jej

přeloží do strojového jazyka konkrétního počítače, čímž z něj prakticky vyrobí v paměti .EXE program [2].

Výsledné programy se v Javě dělí na dvě velké skupiny. První skupinu tvoří aplikace, což jsou běžné programy, druhou skupinu tvoří aplety, které se používají na WWW (World Wide Web) stránkách [1].

1.1.3 Platformy Javy

Než došlo k tomu, že společnost Sun poprvé vydala platformu Java 2 svým zákazníkům, bylo nutné ji nejprve rozdělit na několik částí. Standardní funkcionalita, považovaná za minimální podporu vyžadovanou pro libovolné prostředí Java, je v balíku s názvem J2SE (Java 2 Standard Edition) [1].

Protože rostl zájem o Javu také na podnikové úrovni a v prostředí aplikačních serverů, vydala společnost Sun verzi J2EE (Java 2 Enterprise Edition), která integruje nové technologie jako například servlety, Enterprise JavaBeans a JSP (Java Server Pages) [1].

Požadavky Javy na systémové prostředky se zvyšovaly s každou novou verzí, stejně jako u většiny softwaru. I když má J2SE kořeny v softwaru pro výrobky spotřební elektroniky, vyžaduje příliš mnoho paměti a výkonu procesoru, než aby vytvářela schůdné řešení pro tuto oblast. Zájem však rostl i u menších zařízení a dokonce i u čipových karet, čímž se Java vracela ke svým kořenům. Proto společnost Sun odpověděla vývojem několika platforem Javy, které jsou pro tuto problematiku určeny. Jsou jimi J2ME (Java 2 Micro Edition), JavaCard, EmbeddedJava a PersonalJava. Všechny tyto platformy jsou založené na předchůdci platformy Javy 2 a to na JDK 1.1 (Java Development Kit) a také přistupují jinak k problému redukování platformy, aby se nevyčerpaly dostupné systémové prostředky [1].

Nás bude ze všech těchto platforem zajímat pouze jediná. Blíže se na ni podíváme v následujících podkapitolách.

1.1.4 J2ME a midlet

J2ME je platformou pro malé přístroje, jejímž záměrem je v dohledné době nahradit různé produkty na bázi JDK 1.1 unifikovanějším řešením na bázi Java 2. Na rozdíl od platforem J2SE a J2EE používaných na stolních počítačích, obsahuje mnohem rozmanitější škálu zařízení s mnohem rozdílnějšími schopnostmi, takže pro ně není možné vytvořit jediný softwarový produkt. J2ME proto není jedinou entitou, ale souborem specifikací, které definují určitou část platformy. Každá z nich se hodí pro danou podmnožinu celé kolekce spotřebních zařízení, která spadají do jeho zadání [1].

Podmnožina úplného programovacího prostředí Javy pro určité zařízení se definuje jedním nebo více profily, které rozšiřují základní vlastnosti konfigurace. Konfigurace a profil či profily, které se hodí pro dané zařízení, závisejí na povaze daného hardwaru [1].

1.1.4.1 Konfigurace

Konfigurace je specifikace definující softwarové prostředí pro nějakou škálu zařízení, která je určena sadou charakteristik, na něž se specifikace spoléhá. Obvykle jsou to:

- typ a velikost dostupné paměti
- typ a frekvence procesoru
- typ síťového připojení, které má zařízení k dispozici.

Každá konfigurace se skládá z virtuálního stroje Javy a standardní kolekce javových tříd, které poskytují programovací prostředí pro aplikační software [1].

Konfigurace také obsahuje standardní sadu tříd jazyka Java. Standardní knihovny tříd definované pro konfiguraci (a profily) musí být založeny na knihovnách platformy Java 2. J2ME v současnosti definuje dvě konfigurace.

Konfigurace CLDC (Connected Limited Device Configuration)

Typickou platformou této konfigurace je mobilní telefon, či organizér PDA (Personal Digital Assistant) s přibližně 512kB volné paměti. Proto je CLDC těsně spojena s tzv. bezdrátovou Javou (Wireless Java), která umožňuje uživatelům koupit a stáhnout do mobilního telefonu malé javové aplikace, známé pod pojmem midlety [1].

Konfigurace CDC (Connected Device Configuration)

Zařízení této konfigurace jsou například dražší PDA, chytré telefony, webové telefony, rezidenční brány (residential gateways) a inteligentní doplňková zařízení STB (Set-Top Box) [1].

1.1.4.2 Profily

Profil doplňuje konfiguraci prostřednictvím přidání dalších tříd, které poskytují funkce vhodné pro určitý druh zařízení, nebo pro specifický segment trhu.

Profil MIDP (Mobile Information Device Profile)

Tento profil přidává do CLDC síťové služby, součásti uživatelského rozhraní a úložný prostor. Je zaměřen hlavně na omezený displej a úložné prostředky mobilních telefonů. Proto poskytuje relativně jednoduché uživatelské rozhraní a základní síťové služby na základě HTTP 1.1 (Hypertext Transfer Protocol). MIDP je nejznámějším profilem J2ME, protože tvoří základ bezdrátové Javy. Obě konfigurace J2ME mají jeden či více asociovaných profilů a některé z nich mohou také spoléhat na jiné profily [1].

Profil PDA

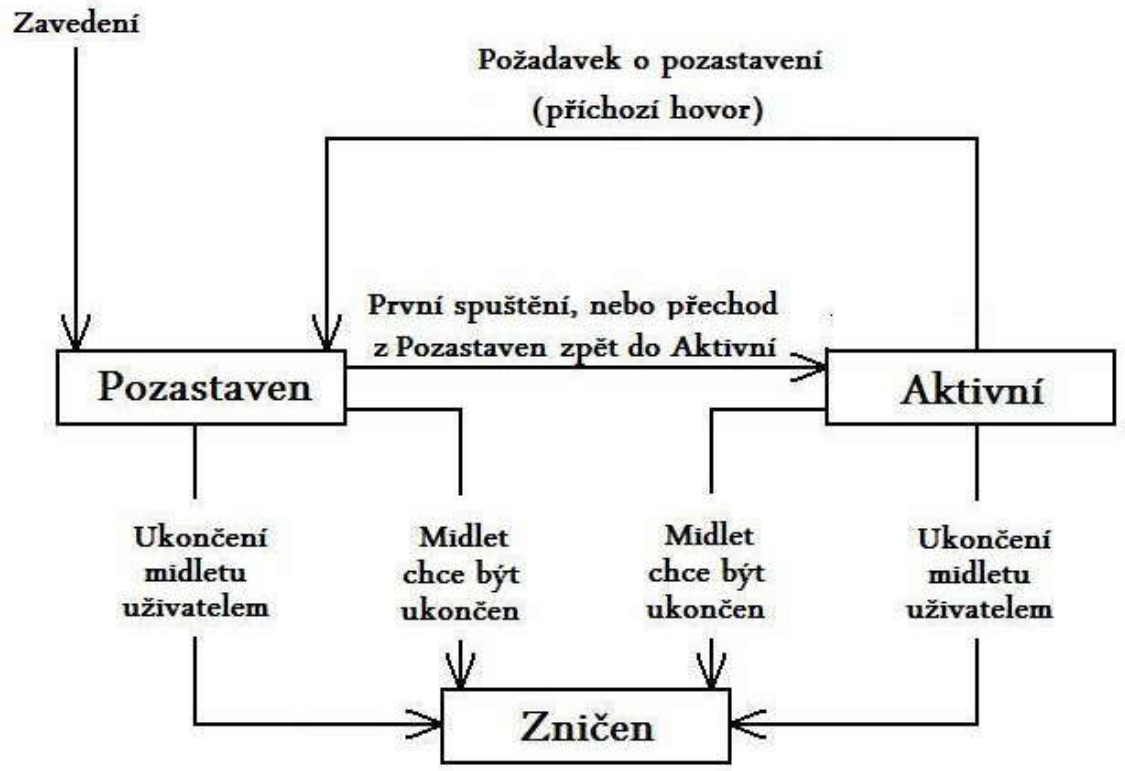
Je podobný MIDP, ale je určen pro organizéry PDA, které mají lepší displeje a více paměti, než mobilní telefony [1].

Ostatní profily

Dalšími profily jsou Základový profil (Foundation Profile), profily Osobní základ (Personal Basis) a Osobní (Personal), dále profil RMI (Remote Method Invocation) a Herní profil. Všechny tyto profily přidávají určité funkce a knihovny. Nás však bude nejvíce zajímat profil MIDP [1].

1.1.4.3 Životní cyklus midletu

Midlet se může nacházet v jednom ze tří stavů. Při zavedení je zpočátku ve stavu Pozastaven. Poté dojde k obvyklé inicializaci tříd a instancí a k zavolání implicitního konstruktoru, a pokud midlet nevyvolá výjimku, je jeho stav změněn na stav Aktivní. Do stavu Pozastaven může platforma MIDP uvést midlet v podstatě kdykoli (například příchozí hovor), přičemž midlet už nemá přístup k obrazovce. Třetím stavem, ve kterém se může midlet nacházet, je stav Zničen, do kterého může být změněn jeho stav také kdykoliv a může být vyvolán buď uživatelem, nebo samotným midletem [1]. Jednoduché znázornění je na obrázku 1.1.



Obr. 1.1: Životní cyklus midletu

1.1.4.4 Síťové možnosti J2ME

Každé MIDP zařízení musí poskytovat jeden komunikační protokol. Jde o docela jednoduchý protokol HTTP. HTTP používá proudové sokety pro přenos zpráv mezi klientem HTTP (obvykle internetový prohlížeč) a webovým serverem, který klientovi často (ale ne vždy) vrací stránku HTML (Hypertext Markup Language). To funguje dobře v prostředí stolních počítačů. U některých mobilních zařízení však žádný prohlížeč není k dispozici. Komunikace tedy proběhne tak, že klient odešle na adresu URL (Uniform Resource Locator) požadavek a serverem je mu vrácena odpověď ve formě sekvence znaků, které musí aplikace sama zpracovat a vyhodnotit [1].

1.1.4.5 Knihovny tříd CLDC

CLDC adresuje celou řadu platforem, které nemají dostatečné paměťové prostředky pro podporu kompletní škály balíčků a tříd z J2SE. Protože je CLDC konfigurací a nikoli profilem, nemá ani žádné volitelné funkce. Knihovna tříd CLDC je velmi malá – je složená z balíčku `javax.microedition.io`, obsahujícího funkce specifické pro J2ME spolu s výběrem tříd z balíčků standardní platformy J2SE:

- `java.io`
- `java.lang`
- `java.util.`

CLDC obsahuje jen omezenou podmnožinu rozsáhlého balíčku `java.io` z J2SE. Jediné vstupní a výstupní proudy, jež lze připojit na skutečný zdroj nebo cíl dat, jsou `ByteArrayInputStream` a `ByteArrayOutputStream`. Tyto proudy mohou mimo jiné poskytovat způsob, jak ukládat a přenášet primitivní datové typy Javy tak, že jsou zabalené pomocí `DataInputStream` a `DataOutputStream`. Balíček `java.lang` obsahuje třídy a rozhraní JVM a balíček `java.util` pak kolekční třídy a třídy týkající se datových a časových údajů. Balíček `javax.microedition.io` obsahuje kolekci rozhraní definujících rámcový systém pro generická připojení. Jeho cílem je, aby ho profily na bázi CLDC využívaly jako společný mechanizmus pro přístup k prostředkům sítě a ostatním prostředkům, jež lze adresovat názvem, a které mohou posílat a přijímat data pomocí `InputStream` a `OutputStream`. Typickým příkladem takového prostředku je stránka HTML či servlet, který lze identifikovat pomocí URL. Pomocí specifikace společných metod nutných k otevírání, zavírání a získávání dat z těchto prostředků, usnadňuje rámcový systém vývojářům psaní aplikací, které mohou být připojeny ke zdrojům dat pomocí různých komunikačních mechanizmů (např. sokety, datagramy nebo HTTP) [1].

1.2 Databáze a jazyk SQL

1.2.1 Databáze a její složení

Databáze (neboli Datová základna) je určitá uspořádaná množina informací (dat) uložená na paměťovém médiu. Tato data slouží pro popis reálného světa (např. evidence studentů). Entita je prvek reálného světa (např. student), který je popsán svými charakteristikami (vlastnostmi). Ty se většinou považují za atribut (např. jméno, příjmení) [4].

Dalším důležitým pojmem je vazba mezi entitami. Jednotlivé entity odpovídající prvkům z reálného světa, mají mezi sebou určitý vztah. Například každý člověk má právě jedny osobní údaje vedené na magistrátu, na oddělení občanských průkazů. To hovoříme o vazbě typu 1:1. Dalším typem je vazba 1:N, jíž bude odpovídat např. skutečnost, že jeden člověk může vlastnit více kreditních karet (ale jedna kreditní karta může být vlastněna pouze jedním člověkem). Posledním typem vazby je M:N. Zde není žádné omezení, příkladem by mohla být situace, že student na vysoké škole si může zapsat několik různých předmětů, ale jeden předmět může být zároveň zapsán více studenty [4].

V širším smyslu jsou součástí databáze i softwarové prostředky, které umožňují manipulaci s uloženými daty a přístup k nim. Tento software se v české odborné literatuře nazývá systém řízení báze dat (SŘBD). Běžně se označením databáze – v závislosti na kontextu – myslí jak uložená data, tak i software (SŘBD) [5].

1.2.2 Relační databáze

Podívejme se nyní blíže na relační model. Základním pojmem je relace**.** Relaci si lze představit jako tabulku, která se skládá ze sloupců a řádků. Sloupce odpovídají jednotlivým vlastnostem (atributům) entity. Údaje v jednom řádku tabulky zobrazují aktuální stav světa.

Mějme tedy tabulku STUDENTI, která bude popisovat entitu studenta vysoké školy. Sloupce tabulky budou ID, JMÉNO, PŘÍJMENÍ, DAT_NAR, uvádějící datum narození studenta a POČ_STUD, informující o započetí studia [4].

ID	JMÉNO	PŘÍJMENÍ	DAT_NAR	FAKULTA	POČ_STUD
72950	František	Kubalík	14.08.85	FEKT	25.9.2006
10021	Petr	Velký	12.10.86	FIT	22.9.2007
76853	Miroslav	Matoušek	01.04.86	FEKT	25.9.2006

Tabulka je základním stavebním kamenem pro budování celé databáze. Relace tedy odpovídá celé tabulce a prvku relace odpovídá jeden konkrétní řádek. Jeden řádek bývá často nazýván databázovým záznamem. Soubor tabulek (relací) pak tvoří celou databázi (relační schéma).

Pojmem hodnota většinou rozumíme uživatelská data. Každý sloupec v tabulce má svůj datový typ (např. celé číslo, řetězec, datum, logická hodnota, apod.). Pro práci s databázovými tabulkami je důležité mít alespoň jednu položku (sloupec), jejíž hodnota nám bude jednoznačně identifikovat záznam v tabulce. Pokud taková položka nebude příliš velká (např. v počtu bajtů), zvolíme ji za tzv. primární klíč. V naší tabulce STUDENTI jím může být položka ID. Primární klíč má tu vlastnost, že jeho hodnota je jedinečná, tj. pro žádné dva řádky v tabulce nemůže nastat situace, že by hodnota primárního klíče byla totožná. Databázové systémy většinou umožňují definovat jako primární klíč n-tici položek, např. dvojici nebo trojici položek. V takovém případě se mohou některé položky v klíčích opakovat, ale nesmí být shodné všechny položky dvou primárních klíčů najednou.

Dalším důležitým pojmem je funkční závislost. Když například v naší tabulce budeme mít sloupec STUD_PRUM znázorňující studijní průměr studenta a sloupec STIPENDIUM, tak by výše stipendia závisela na velikosti průměru studenta a zapisovalo by se to takto: STUD_PRUM->STIPENDIUM .

1.2.2.1 Normální formy

Pojem normálních forem se používá ve spojitosti s dobře navrženými tabulkami. Správně vytvořené tabulky splňují 4 základní normální formy [4].

1. normální forma (1NF)

První, nejjednodušší normální forma říká, že všechny atributy jsou atomické, tj. dále již nedělitelné (jinými slovy, hodnotou nesmí být relace).

Mějme například další tabulku ADRESY, která bude mít sloupce JMÉNO, PŘÍJMENÍ a BYDLIŠTĚ.

JMÉNO	PŘÍJMENÍ	BYDLIŠTĚ
František	Kubalík	Vedlejší 5, Havířov 73601
Petr	Velký	Školní 25, Havířov 73601
Miroslav	Matoušek	Štefánkova 12, Ostrava 79002

Pokud bychom v této tabulce chtěli vypsat všechny studenty, jejichž PSČ je rovno určité hodnotě, nemohli bychom ho jednoduše zjistit, protože vlastnost BYDLIŠTĚ není atomická. Správný návrh tabulky respektující 1NF bude vypadat takto:

JMÉNO	PŘÍJMENÍ	ULICE	ČÍSLO	MĚSTO	PSČ
František	Kubalík	Vedlejší	5	Havířov	73601
Petr	Velký	Školní	25	Brno	73601
Miroslav	Matoušek	Štefánkova	12	Ostrava	79002

Musíme se tedy snažit, aby obsahem jedné databázové položky byla právě jedna hodnota (určitého databázového typu).

2. normální forma (2NF)

Tabulka splňuje 2NF, právě když splňuje 1NF a navíc každý atribut, který není primárním klíčem, je na primárním klíči úplně závislý. To znamená, že se nesmí v řádku tabulky objevit položka, která by byla závislá jen na části primárního klíče. Z definice vyplývá, že problém 2NF se týká jenom tabulek, kde volíme za primární klíč více položek než jednu. Jinými slovy, pokud má tabulka jako primární klíč jenom jeden sloupec, pak 2NF je splněna triviálně. Mějme tabulku ZÁPISY, která bude vypadat následovně (atribut ID_PŘED značí číslo předmětu, který má daný student zapsán):

ID	JMÉNO	PŘÍJMENÍ	ID_PŘED	PŘEDMĚT
72950	František	Kubalík	12	BAN2
10021	Petr	Velký	12	BAN2
76853	Miroslav	Matoušek	3	BCIF

Pokud zvolíme jako primární klíč pouze ID, je to špatně, neboť zcela určitě název předmětu, který má student zapsán, není závislý na číslu studenta, takže za primární klíč musíme vzít dvojici (ID, ID_PŘED). Tím ovšem vznikl nový problém. Položky JMÉNO, PŘÍJMENÍ a PŘEDMĚT nejsou úplně závislé na dvojici zvoleného primární klíče. Musíme tedy provést "rozpad" na více tabulek, což se odborně označuje dekompozice relačního schématu. Správně navržené tabulky splňující 2NF budou vypadat následovně (tabulka ZÁPISY a PŘEDMĚTY):

ID	JMÉNO	PŘÍJMENÍ	ID_PŘED
72950	František	Kubalík	12
10021	Petr	Velký	12
76853	Miroslav	Matoušek	3

ID_PŘED	PŘEDMĚT
12	BAN2
3	BCIF

V původní tabulce nám došlo k jevu, který nazýváme redundance. To proto, že se nám tam vyskytovala dvakrát informace, že předmět s číslem 12 se nazývá BAN2. Tento jev často doprovází nesplnění 2NF a je nežádoucí.

3. normální forma (3NF)

Relační tabulky splňují třetí normální formu, jestliže splňují 2NF a žádný atribut, který není primárním klíčem, není tranzitivně závislý na žádném klíči. Mějme tabulku STIPENDIA, která bude vypadat takto:

ID	JMÉNO	PŘÍJMENÍ	STUD_PRUM	STIPENDIUM
72950	František	Kubalík	1.500	900
10021	Petr	Velký	2.410	0
76853	Miroslav	Matoušek	1.210	1300

Tato tabulka zatím nesplňuje ani 2NF, což je první předpoklad pro splnění 3NF. Můžeme vidět, že JMÉNO, PŘÍJMENÍ a STUD_PRUM závisí na atributu ID (ten by byl pravděpodobně primárním klíčem). Dále můžeme vidět, že atribut STIPENDIUM je zřejmě funkčně závislý na atributu STUD_PRUM a pokud vezmeme v úvahu, že ID->STUD_PRUM a STUD_PRUM->STIPENDIUM, dostaneme díky jevu zvanému tranzitivita, že ID>STIPENDIUM. Postup, jak dostat tabulky do 3NF, je podobný jako v případě 2NF, tj. opět provedeme dekompozici (tabulka PROSPĚCHY a STIPENDIA):

ID	JMÉNO	PŘÍJMENÍ	ID_PRUM
72950	František	Kubalík	3
10021	Petr	Velký	4
76853	Miroslav	Matoušek	1

ID	ROZSAH	STIPENDIUM
1	<1;1,25)	1300
2	<1,25;1,4)	1100
3	<1,4;1,55)	900
4	<1,55;1,7)	0

Takto navržené tabulky už jsou v pořádku. Pro splnění podmínky, že primární klíč tabulky má být nejkratší délky, jsme zavedli ještě jeden sloupec pro číslování jednotlivých rozsahů pro stipendia.

Boyce-Coddova normální forma (BCNF)

Toto je poslední prakticky užívaná forma. Tabulka splňuje BCNF, právě když pro dvě množiny atributů A a B platí: A->B a současně B není podmnožinou A. Pak množina A obsahuje primární klíč tabulky. Tato forma zjednodušuje práci s tabulkami. Ve většině případů, pokud dobře postupujeme při tvorbě tabulek, aby splňovaly postupně 1NF, 2NF a 3NF, forma BCNF je splněna.

1.2.3 Jazyk SQL

1.2.3.1 Struktura jazyka

Tento jazyk v sobě zahrnuje nástroje pro tvorbu databází (tabulek) a dále nástroje na manipulaci s daty (vkládání dat, aktualizace, mazání a vyhledávání informací). Patří mezi tzv. deklarativní programovací jazyky, což v praxi znamená, že kód jazyka nepíšeme v žádném samostatném programu (jako by tomu bylo např. u jazyka C nebo Pascal), ale vkládáme jej do jiného programovacího jazyka, který je již procedurální. Se samotným jazykem SQL můžeme pracovat pouze v případě, že se terminálem připojíme na SQL server a na příkazový řádek zadáváme přímo příkazy jazyka [4].

SQL se skládá z několika částí. Některé části jsou určeny pro administrátory a návrháře databázových systémů, jiné pak pro koncové uživatele a programátory. První částí jazyka SQL je jazyk DDL (Data Definition Language). Jedná se o jazyk pro vytváření databázových schémat a katalogů. Způsob ukládání tabulek definuje jazyk SDL (Storage Definition Language). Třetí částí pro návrháře a správce je jazyk VDL (View Definition Language), určující vytváření pohledů. Pohled si lze představit jako virtuální tabulku složenou z různých jiných tabulek. Poslední částí je jazyk DML (Data Manipulation Language), který obsahuje základní příkazy INSERT, UPDATE, DELETE a nejpoužívanější příkaz SELECT [4]. S jazykem DML pracují nejvíce koncoví uživatelé a programátoři databázových aplikací a právě s ním budeme pracovat i my.

1.2.3.2 Databázový server MySQL

MySQL je v podstatě relační databázový server. Už od prvního vydání tohoto softwaru kladli tvůrci MySQL značný důraz na výkon a škálovatelnost (scalability). To vedlo k vysoce optimalizovanému produktu, který ovšem postrádal mnohé schopnosti, které se považují za standardní u databázových produktů určených pro velké korporace: například uložené procedury, triggery a transakce. Přesto k sobě produkt přitáhl pozornost nesmírného množství uživatelů, které spíše zajímala rychlost a zvětšitelnost než atraktivní výbava, která v mnoha případech stejně zůstává stranou, nevyužitá. S každou další vydanou verzí přicházely dodatečné schopnosti, které přilákaly další nové uživatele. Díky svým vlastnostem je velice populární. Jsou to [3]:

Flexibilita

Výkon:

- Schopnosti SQL na úrovni velkých korporací
- Fulltextové indexování a vyhledávání
- Ukládání dotazů do čachr
- Replikace
- Bezpečnost

Flexibilní licenční možnosti:

- Licence MySQL Open Source
- Komerční licence

Hyperaktivní komunita uživatelů

1.3 Skriptovací jazyk PHP

1.3.1 Historie

Počátky PHP se datují kolem roku 1995, kdy jistý Rasmus Lerdorf vyvinul skript, který protokoloval informace o návštěvnících a zobrazoval na webové stránce počet návštěvníků. Tím vznikl jeden z nejpopulárnějších skriptovacích jazyků na celém světě. Postupem času se vydávaly stále zdokonalující se verze. Vydáním verze PHP 5 dosáhla jeho obliba historického maxima [3].

1.3.2 Všeobecné rysy jazyka

1.3.2.1 Upotřebitelnost

Původní myšlenkou nebylo vytvořit nový jazyk, ale vyřešit konkrétní úkol, pro který právě neexistovalo použitelné řešení. Prospěšný skript tedy může tvořit klidně jen jediný řádek, na kterém může být do sebe vnořeno několik funkcí. Není zde také nutné explicitně vytvářet, přetypovávat, nebo likvidovat proměnnou. Vše se děje automaticky.

1.3.2.2 Vyspělost

Za dobu svého vývoje urazil jazyk PHP pěkný kus cesty. V současné době je dostupných 113 knihoven, které mají dohromady dost přes 1000 funkcí. Díky tomu můžeme např.:

- komunikovat s databází
- manipulovat s informacemi z formulářů
- zanalyzovat i ty nejsložitější řetězce pomocí knihoven regulérních výrazů
- vytvářet stránky dynamicky
- a mnoho dalšího.

1.3.2.3 Možnosti

Vývojáři PHP jsou jen výjimečně svázáni jediným možným implementačním řešením. Naopak, uživatel může být snadno zahlcen spoustou voleb, které jazyk nabízí. Pokud například zvážíme možnosti PHP na poli databází, nabízí se nám podpora pro více než dvacet pět databázových produktů. Vyspělé schopnosti PHP pro analýzu řetězců jsou dalším důkazem svědčícím o možnostech nabízených uživatelům.

2 POPIS REALIZACE A TESTOVÁNÍ

2.1 Cíl

Mým cílem bylo navrhnout projekt, kde bude mobilní terminál komunikovat s databází. Realizace proto má několik kroků, které bylo nutno projít, a to od vhodného návrhu tabulek, až po realizaci samotné komunikace mezi databází a aplikací (midletem) běžící v mobilním terminálu.

2.2 Realizace

2.2.1 Návrh databáze

Jak již bylo řečeno v teoretickém úvodu, tabulka je základním stavebním kamenem celé databáze. Návrh tabulek proto musí být pečlivý a promyšlený. Prvním úkolem bylo navrhnutí centrální databáze, na které závisí běh celé aplikace (midletu) [7].

2.2.1.1 Tabulky

Tabulka UZIVATEL

Aplikace má mít za úkol umožnit uživateli sázet na sportovní zápasy. Z toho jsou zřejmé první dvě tabulky, které budou potřeba. První tabulka byla nazvána UZIVATEL, popisující entitu uživatele sázkové společnosti. Sloupce tabulky jsou ID_UZIVATELE pro identifikační číslo uživatele, LOGIN a HESLO, pod kterým se pak bude uživatel k sázkové kanceláři přihlašovat, dále PRIJMENI, JMENO, ULICE, CISLO_POPISNE, MĚSTO, PSC, EMAIL, TELEFON, DATUM_NAROZENI, a poslední KONTO, udávající dostupný kredit uživatele. Jako primární klíč slouží sloupec ID. Jako ukázka je uvedena tato tabulka jen s důležitými sloupci. S jedním uživatelem, se kterým byla později aplikace testována, tedy tabulka vypadá takto:

id_uzivatele	login	heslo	konto
35	ferda	ferda	11461.5

Tabulka ZAPAS

Další tabulka byla nazvána ZAPAS. Ta popisuje entitu zápasu nabízeného sázkovou společností ke vsazení. Sloupce tabulky jsou ID_ZAPASU pro identifikační číslo zápasu, ID_SOUTEZE, což je cizí klíč ke třetí tabulce SOUTEZ (viz níže), CISLO_ZAPASU, což je číslo které bude uživatel při sázení na konkrétní zápas zadávat, dalšími sloupci jsou sloupce DOMACI, HOSTE a sloupce pro kurzy na jednotlivé tipy, tedy VD (výhra domácích), R (remíza) a VH (výhra hostů), dále sloupec VYSLEDEK pro informaci, jak zápas skončil. Tento

sloupec bude nabývat hodnot 1, 2 nebo 3, kdy hodnota 1 znázorňuje výhru domácích, hodnota 2 výhru hostů a hodnota 3 nerozhodný stav. Předposlední sloupec je sloupec DATUMCAS a poslední sloupec této tabulky je nazván STAV. Tento sloupec nám říká, zda je zápas odehrán (hodnota 3), nebo ne (hodnota 1). Hodnota 2 znázorňuje, že se zápas už hraje. Tato hodnota ovšem není kvůli testování brána v úvahu, stejně jako sloupec DATUMCAS, na kterém by tato hodnota závisela. Jako primární klíč je opět zvolen sloupec ID. Tabulka (opět jsou vybrány jen důležité sloupce) s jedním údajem tedy vypadá takto:

id_zapasu	id_souteze	cislo_zapasu	domaci	hoste	1	0	2	vysledek	stav
12	7	9	Znojmo	Zlin	1.5	2.2	2.1	NULL	1

Tabulka SOUTEZ

Nyní se vraťme k již dříve zmiňované tabulce SOUTEZ, na niž bylo odkazováno prostřednictvím cizího klíče z předcházející tabulky. Tato tabulka má pouze informativní charakter. Obsahuje sloupce ID_SOUTEZE (primární klíč), ZEME, SPORT, LIGA a ZKRATKA, přičemž posledně jmenovaným údajem bude uživatel o soutěži kam konkrétní zápas patří dostatečně informován. Příklad tabulky se třemi údaji:

id_souteze	zeme	sport	liga	zkratka
5	Ceska republika	Fotbal	1.liga	Ces1L
7	Ceska republika	Hokej	Extraliga	HCesE
9	Recko	Fotbal	1.liga	Rec1L

Tabulka ROZPRACOVANE_TIKETY

Další tabulka, která bude potřeba, je nazvána ROZPRACOVANE_TIKETY. Tato tabulka má pouze dva sloupce. Jsou jimi ID_ROZPR_TIKETU, což je primární klíč a ID_UZIVATELE, plnící funkci cizího klíče k tabulce UZIVATEL. Do této tabulky budou automaticky ukládána identifikační čísla všech uživatelů při jejich přihlášení a zase mazána při jejich odhlášení. Příklad tabulky:

id_rozpr_tiketu	id_uzivatele
69	35

Použitím této tabulky bude mít každý uživatel přidělen právě jeden dočasný rozpracovaný tiket, do kterého budou ukládány zápasy prostřednictvím další tabulky, nazvané ZAPASY_NA_ROZPR_TIKETU.

Tabulka ZAPASY_NA_ROZPR_TIKETU

Tabulka obsahuje potřebné údaje všech zápasů, které si uživatel dočasně zadal na rozpracovaném tiketu. Dočasně proto, že si zde uživatel může zápasy přidávat zápasy dle libosti, stejně jako odstraňovat. Tabulka obsahuje sloupec ID_ZAP_ROZPR_TIK plnící funkci primárního klíče, ten je však pro nás nezajímavý. Naopak důležité jsou další sloupce ID_ZAPASU, plnící funkci cizího klíče k tabulce ZAPAS, ID_ROZPR_TIKETU, což je cizí klíč k tabulce ROZPRACOVANE_TIKETY, sloupec TIP informuje, pro jaký tip se uživatel rozhodl a sloupec KURZ informuje o velikosti kurzu pro daný tip. Touto kombinací cizích klíčů je zajištěno to, že budeme vědět, na kterém rozpracovaném tiketu se nachází jaký zápas. Příklad tabulky je následující:

id_zap_rozpr_tik	id_zapasu	id_rozpr_tiketu	tip	kurz
56	12	69	1	1.5

Tabulka TIKET

Předpokládejme, že se uživatel rozhodne rozpracovaný tiket vsadit. Je tedy potřeba uložit celý jeho obsah do další tabulky, která už nebude dočasná. Nazývá se TIKET. První dva sloupce má stejné jako tabulka ROZPRACOVANE_TIKETY, ale obsahuje pár sloupců navíc. Jsou jimi SAZKA, MAX_KURZ, udávající součin všech kurzů tipů, na které uživatel vsadil, dále sloupec VYHRA, který bude vlastně obsahovat součin maximálního kurzu a sázky a sloupce ROZHODNUTO, SPRAVNOST, DATUMCAS a STAV. Všechny sloupce mají pouze informativní charakter, poslední sloupec STAV bude hrát velkou roli při kontrole správnosti tiketu. Hodnota 1 znamená nerozhodnutý tiket, hodnota 2 nevyhrávající tiket, hodnota 3 vyhrávající tiket a hodnota 4 znamená, že tiket již byl prohlédnut uživatelem. Zde je příklad tabulky, sloupec DATUMCAS je záměrně vynechán:

id_tiketu	id_uzivatele	sazka	max_kurz	vyhra	rozhodnuto	spravnost	stav
4	35	20	5.25	105.00	ANO	ANO	4

Tabulka ZAPASY_NA_ TIKETU

Předposlední tabulka je nazvána ZAPASY_NA_TIKETU. Ta popisuje entitu zápasu, který se nachází na určitém tiketu. Bude tedy obsahovat sloupce ID (primární klíč), ID_ZAP_TIK, ID_ZAPASU, ID_TIKETU, TIP, KURZ, SPRAVNOST a STAV. Všechny sloupce plní stejnou funkci jako v tabulce ZAPASY_NA_ROZPR_TIKETU, sloupec SPRAVNOST je zde navíc. Jeho funkce je jistě zřejmá a je pouze informativní. Poslední sloupec STAV je velmi důležitý při kontrole správnosti zápasů, kdy hodnota 1 znamená nerozhodnutý zápas, hodnota 2 špatně

tipnutý zápas a hodnota 3 správně tipnutý zápas. I když v této tabulce sloupec ID pravděpodobně nebudeme potřebovat, musíme jej zde uvést pro primární klíč. Tabulka bude vypadat tedy takto:

id_zap_tik	id_zapasu	id_tiketu	tip	kurz	spravnost	stav
9	5	4	0	3.5	ANO	3

Tabulka BETMOBILE_PRIHLASENI

Tato poslední tabulka je dočasná a plní důležitou funkci, která bude popsána později. Obsahuje údaje o přihlášených uživatelích prostřednictvím aplikace běžící na mobilním terminálu. Skládá se ze sloupců ID_PRIHL_UZIVATELE, ID_UZIVATELE, LOGIN, HESLO. Údaje jsou z ní vymazány po odhlášení uživatele z aplikace.

2.2.2 Realizace midletu

Import balíčků

Midlet, starající se o komunikaci s uživatelem, byl pracovně nazván BetMobileMidlet. Pro správnou funkci všech metod byla potřeba importovat potřebné knihovny (balíčky):

- `java.io.*`
- `javax.microedition.midlet.*`
- `javax.microedition.lcdui.*`
- `javax.microedition.io.*.`

Deklarace proměnných

Poté následovala deklarace všech proměnných, které byly v programu použity, tzn. proměnná pro obsluhu displeje, proměnné pro tvorbu příkazů, proměnné pro tvorbu formulářů, jedna proměnná pro běžící text, dále proměnné pro tvorbu textových polí, kam uživatel zadává data, obyčejné řetězcové proměnné pro přenášení potřebných dat v URL (pro reprezentaci byla taky vytvořena jedna proměnná) směrem ke skriptu a proměnné primitivních datových typů `int` (pouze pro indikaci zobrazovaných údajů na displeji) a `char` (pouze pro korektní zobrazování výsledků skriptu na displeji). Jako poslední byla deklarována proměnná `RunScript`, prostřednictvím které se bude volat skript v novém vlákně. K tomu pomáhá implementace rozhraní `Runnable`. Zde je uvedena deklarace příkazu Prihlasit:

```
private Command connect = new Command("Prihlasit", Command.SCREEN, 1);
```

Pomocí klíčového slova `new` je zde vytvořen objekt třídy `Command`. První argument je zobrazovaný název, druhý argument značí, že se jedná o příkaz týkající se funkcí aktuální obrazovky. Příkaz pro ukončení by zde měl `Command.EXIT`. Poslední argument označuje prioritu zobrazení příkazu v nabídce, příkaz pro ukončení se opět odlišuje prioritou 0.

Metody

Po spuštění midletu je zavolána metoda `startApp()`. Do té byla implementována metoda `displayLogin()`, která uživateli zobrazí na displej potřebná textová pole a přidá do nabídky potřebné příkazy. Zde je ukázka:

```
public void displayLogin()
{
 loginForm = new Form("Prihlasit k BetMobile");
 ticker = new Ticker("Zadejte Login a Heslo ");
 tf_login = new TextField("Login: ","",30,TextField.ANY );
 tf_password = new TextField("Heslo: ","",30,TextField.PASSWORD);
 loginForm.setTicker(ticker);
 loginForm.append(tf_login);
 loginForm.append(tf_password);
 loginForm.addCommand(exit);
 loginForm.addCommand(connect);
 loginForm.setCommandListener(this);
 display.setCurrent(loginForm);
}
```

Při vytváření textových polí tvoří první argument popisek textového pole, následuje počáteční hodnota, poté maximální počet zadávaných znaků a omezení, neboli formát zadávaných dat. Každá položka, která se má na formuláři zobrazit, se musí připojit pomocí metody `append()`, příkaz pomocí metody `addCommand()`. Metoda `setCommandListener()` poté nastavuje posluchač, aby midlet věděl, o vykonání kterého příkazu uživatel požádal. Pomocí metody `setCurrent()` se pak zobrazí celý formulář na displeji.

Hlavní formulář byl vytvořen obdobně. Jsou zde samozřejmě změněny příkazy, které se obměňují podle právě zobrazené obrazovky, stejně jako hlavičky tiketů, nabídky zápasů, atd. Totéž platí pro zobrazovaná textová pole. Těchto obměn je docíleno pomocí podmínek ověřujících stav k tomu určených proměnných.

Jak již bylo řečeno výše, pro detekci zvoleného příkazu se musel nastavit posluchač. Při aktivaci jakéhokoliv příkazu se tedy tato operace provedla a zavolala se metoda `commandAction()`. Jeden z argumentů této metody midletu říká, který konkrétní příkaz byl aktivován. Na základě toho se pak pomocí podmínek uvnitř této metody zjistí, co se má provést. Následuje ukázka z metody `commandAction()` s příkazem Nabidka zapasu (název proměnné pro tento příkaz je `matches`):

```
else if (command == matches)
{
 commands = 3;
 showMatchDataBox = 1;
 showHeadMatch = 1;
 url = "http://localhost:80/BetMobileScripts/betmobile.php?action=matches";
 script = new RunScript(this);
 script.start();
}
```

Stav proměnných `commands`, `showMatchDataBox` a `showHeadMatch` určuje, že se mají zobrazit určité příkazy, hlavička nabídky zápasů a textová pole pro zadání čísla zápasu a tipu. Adresa URL se skládá z názvu protokolu, hostitele, který je v tomto případě `localhost`, čísla portu, cestě ke skriptu, který má být zavolán a z parametrů. Parametry začínají znakem `?` a skládají se ze sady dvojic `název=hodnota`. V některých URL těchto dvojic může být víc, ty jsou pak odděleny znaky ampersand (&). Například pro vsazení tiketu je do proměnné `url` uložen tento řetězec:

```
"http://localhost:80/BetMobileScripts/betmobile.php?action=betTicket&loginU
ser="+loginUser+"&bettedMoney="+bettedMoney
```

Znaky `+` jsou odděleny řetězcové proměnné, ve kterých jsou uloženy potřebné informace od uživatele. Poslední dva řádky v ukázce se starají o vytvoření objektu vytvořené třídy `RunScript` a o spuštění nového vlákna metodou `start()`. Konstrukce pro ostatní příkazy v metodě `commandAction()` je obdobná.

Nakonec zbývá popsat konstrukci již několikrát zmiňované třídy `RunScript`. Mimo metody `start()` obsahuje také metodu `run()`, která otevírá připojení k URL a ukládá výsledek volaného skriptu. Zde je část kódu této metody:

```
HttpConnection c = (HttpConnection) Connector.open(url);
c.setRequestMethod(HttpConnection.POST);
DataInputStream is =(DataInputStream)c.openDataInputStream();
int ch;
sb = new StringBuffer();
if(showLoginForm == 1)
{
 showLoginForm = 0;
 display.setCurrent(loginForm);
}
else
{
 while ((ch = is.read()) != -1)
 {
  checkChar = (char)ch;
  if(checkChar != '#')
  {
   sb.append(checkChar);
  }
  else
  {
   result = sb.toString();
   mainForm.append(result);
   display.setCurrent(mainForm);
   sb.delete(0,sb.length());
  }
 }
}
is.close();
c.close();
```

Po otevření spojení a nastavení metody je potřeba ještě deklarovat vstupní proud pro data zvaný `DataInputStream` a tento proud otevřít. Do řetězcového bufferu bude ukládána přijímaná informace. Ještě před tím je zde ale podmínka, která ověřuje, jestli se náhodou uživatel neodhlásil a není potřeba zobrazit přihlašovací formulář. Pokud je vše v pořádku, probíhá pomocí cyklu `while()` načítání informací vrácených skriptem znak po znaku. Pro kontrolu konce načítaného řádku je zde znak `#`. Tento znak automaticky přidává zavolaný skript. V momentě, kdy je tedy tento znak načten, řetězec je zobrazen na displej a pokračuje se v načítání dalšího řetězce. V opačném případě se jen k bufferu přidávají další znaky. Po ukončení načítání, tzn. `ch = is.read()) == -1`, se uzavře vstupní proud a spojení http [8].

2.2.3 Realizace skriptu

O veškerou komunikaci s databází se stará jeden rozsáhlý skript PHP. Je rozdělen na několik částí, přičemž ze všeho nejdříve je potřeba připojit se k databázi. Po úspěšném připojení se pak pomocí podmínek ověřuje, jaké parametry URL obsahuje. Podle těchto parametrů se pak ve skriptu skočí na určitý blok kódu a ten se pak vykonává. Načtení parametru je zde docíleno pomocí funkce `GET`.

Nyní je na čase se vrátit k tabulce BETMOBILE_PRIHLASENI. Po odeslání uživatelského loginu a hesla v midletu, jsou tyto informace přijaty skriptem. Ten se podle nich pokusí vybrat data z databáze a v případě úspěchu informuje midlet hláškou. Aby se vědělo, že je uživatel úspěšně přihlášen, musí se do tabulky BETMOBILE_PRIHLASENI uložit potřebná data o uživateli. Potom už se jen ověřuje, zda má uživatel, který žádá o provedení nějaké operace (v URL je vždy uložen login uživatele), záznam v této tabulce. Při odhlášení se samozřejmě data daného uživatele z této tabulky vymažou.

Jako ukázka ze skriptu byla zvolena operace přepočtení výhry na rozpracovaném tiketu, kde má uživatel zadány nějaké zápasy a chce jen zjistit, jakou částku by vyhrál v případě správnosti tiketu:

```
elseif($_GET["action"] == showWin)
{
 $maxCourse = 1;
 $bettedMoney = $_GET["bettedMoney"];
 $win = 0;
 $loginUser =  $_GET["loginUser"];
 $query = "SELECT * FROM betmobile_prihlaseni WHERE login = '$loginUser'";
 $result = mysql_query($query);
 if (mysql_num_rows($result) == 0)
 {
  echo "Potize pri overeni uzivatele. Prihlaste se prosim znovu.#";
 }
 else
 {
```

```
  $userData = mysql_fetch_assoc($result);
  $query="SELECT * FROM zapasy_na_rozpr_tiketu zrt, rozpracovane_tikety rt
  WHERE rt.id_uzivatele='".$userData["id_uzivatele"]."'AND
  zrt.id_rozpr_tiketu = rt.id_rozpr_tiketu";
  $resultCurrentTicket = mysql_query($query);
  if(mysql_num_rows($resultCurrentTicket) == 0)
  {
   echo "V tiketu nejsou zadne zapasy!!!#";
  }
  else
  {
   while($ticketData = mysql_fetch_assoc($resultCurrentTicket))
   {
    $maxCourse *= $ticketData["kurz"];
   }
   $win = $maxCourse * $bettedMoney;
   echo "Rozpracovany tiket s max. kurzem ".$maxCourse." ma se zadanou
   ".$bettedMoney." Kc vyhru ".$win." Kc.#";
  }
 }
}
```

Při shodnosti parametru v URL s první podmínkou, se do proměnné `$bettedMoney` načte opět z URL parametr `bettedMoney`. V tom je uložena částka, kterou zadal uživatel v midletu a kterou chce přepočítat na výhru. Jsou nastaveny potřebné proměnné a pro ověření uživatele načten poslední parametr `loginUser` do proměnné `$loginUser`. Pokud se tento login vyskytuje v tabulce BETMOBILE_PRIHLASENI, je vše v pořádku a uživatelská data jsou pro další použití z této tabulky načtena do asociativního pole `$userData`. Dalším krokem je načtení údajů z rozpracovaného tiketu. Při postupném načítání zápasů se do proměnné `$maxCourse` ukládá maximální kurz tiketu (vynásobení všech kurzů na tiketu). Poté už je jen do proměnné `$win` uložena možná výhra (pouze vynásobení maximálního kurzu a sázky) a zobrazen výsledek, který si přebere midlet.

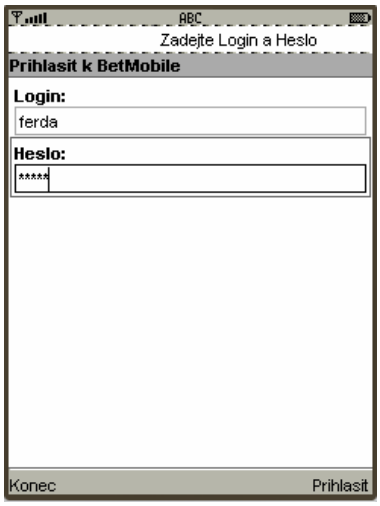
Všechny další operace, ať už zobrazování nabídky zápasů, zobrazování všech tiketů uživatele, nebo jen zjišťování výše konta klienta jsou řešeny obdobně. Liší se pouze adresa URL a samozřejmě SQL dotazy a stav proměnných. Vše tedy provádí skript a to včetně kontroly tiketu a přičítání výher ke kontu klienta [6].

2.3 Testování

Databáze byla vyvinuta na databázovém serveru MySQL: 5.0.51b pomocí nástroje phpMyAdmin verze 3.0.1.1 a skript PHP byl psán v textovém editoru Adobe Dreamweaver CS4. Midlet byl vyvinut v prostředí NetBeans IDE 6.1 [9], kde byl i testován na výchozím emulátoru mobilního telefonu. Celá aplikace byla testována jak pro server localhost, který byl spuštěn přímo v počítači, tak pro webovou stránku *http://betmobile.own.cz*, kam byl skript obsluhující midlet nahrán. Obě varianty vykazovaly stejné výsledky.

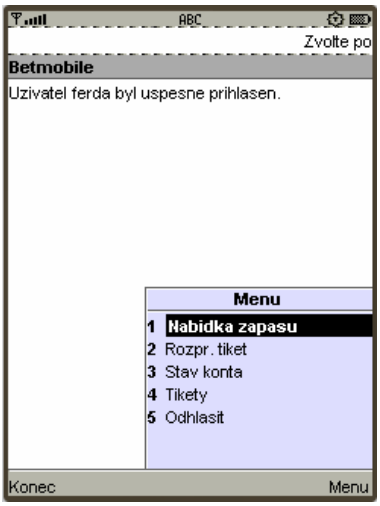
2.3.1 Přihlášení, zobrazení nabídky a zjištění konta

Po spuštění midletu nás uvítá přihlašovací obrazovka se dvěma textovými poli a běžícím textem (obr.2.3a). Po zadání přihlašovacích údajů a jejich odeslání se nám zobrazí hláška o úspěšnosti, viz obr.2b nebo obr.2c. Nepřihlášený uživatel má sice stejné menu jako uživatel přihlášený, jeho snaha o provedení nějakého příkazu (kromě zobrazení nabídky zápasů) však bude zastavena chybovou hláškou (např. při zjišťování kreditu, viz obr.2.3d). Po úspěšném přihlášení se naplní tabulky BETMOBILE_PRIHLASENI a ROZPRACOVANE_TIKETY (obr.2.3g) a uživatel může zjistit svůj kredit (obr. 2.3e), nebo zobrazit nabídku zápasů (obr. 2.3f).



Obr. 2.3a

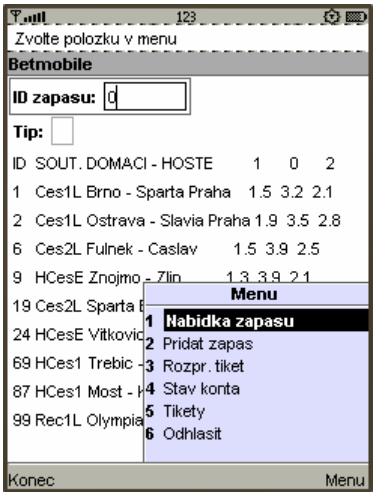
Obr. 2.3b



Obr. 2.3c

Obr. 2.3d

Obr. 2.3e



Obr. 2.3f

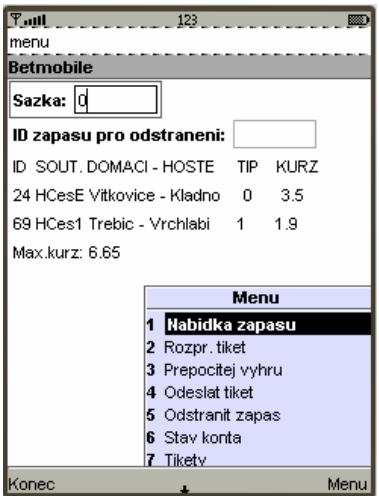
Obr. 2.3g

V případě, že by přihlášený uživatel zadal příkaz *Rozpr.tiket* nebo *Tikety,* byla by mu zobrazena chybová hláška *V tiketu nejsou zadne zapasy*!!!, ve druhém případě *Zadne tikety nenalezeny!!!*.

2.3.2 Přidávání zápasu, odebírání zápasu, přepočítání možné výhry

Vycházejme ze skutečnosti, že si uživatel chce do rozpracovaného tiketu přidat zápas. Do kolonky *ID zápasu* a *Tip* tedy zadá požadované údaje a zvolí položku *Pridat zapas.* V případě správného zadání *ID zápasu* a *Tipu* je zápas přidán do tiketu a uživatel je o tom informován na displeji hláškou (např. pro zápas s ID 24) *Zapas cislo 24 byl uspesne pridan.* Pokud byly údaje zadány nekorektně, na displeji se zobrazí chybové hlášky ve znění *Udaj "ID zapasu" byl nekorektne zadan!!!* nebo *Udaj "Tip" byl nekorektne zadan!!!*

Na obr. 2.3h je už zobrazen rozpracovaný tiket se dvěma zápasy. Uživatel může v případě potřeby odstranit zápas z tiketu zadáním údaje do kolonky *ID zapasu pro odstraneni* a zvolením nabídky *Odstranit zapas*. Poté je opět informován o úspěšnosti hláškami *Udaj "ID zapasu" byl nekorektne zadan!!!* nebo *Zapas byl uspesne odebran z rozpracovaneho tiketu.* Další možnost, kterou uživatel má, je přepočítání možné výhry, kdyby tiket vsadil za zadanou částku do kolonky *Sazka.* Tato kolonka je omezena pouze na čísla, takže ošetření vstupu není třeba. Po zadání částky a zvolení položky *Prepocitej vyhru* se nám tedy na displeji zobrazí požadovaná informace (obr. 2.3i).



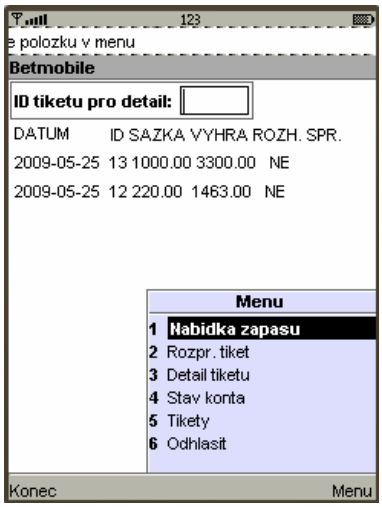
Obr. 2.3h

Obr. 2.3i

2.3.3 Odeslání tiketu, zobrazení všech tiketů, zobrazení detailů tiketu

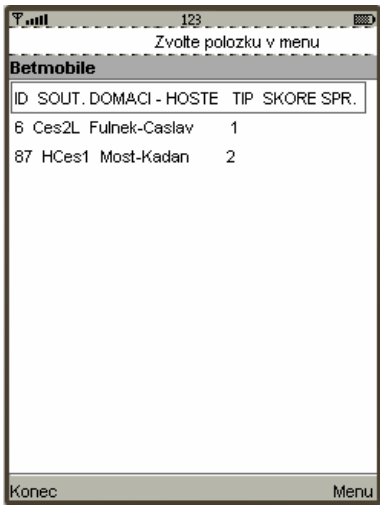
V předchozí podkapitole jsme tedy sestavili tiket, který konečně můžeme odeslat ke vsazení. Tomu musí samozřejmě předcházet zadání sázky. Po všech potřebných kontrolách jako jsou ověření duplicity zápasů na tiketu, omezení maximální výhry, dostatečný kredit uživatele a

ověření nenulové sázky (všechny chybové hlášky jsou v případě potřeby zobrazeny na displeji), je po vybrání položky *Odeslat tiket* tiket vsazen (opět je zobrazena hláška informující o úspěchu operace) a zápasy z rozpracovaného tiketu jsou vymazány. Nyní si může uživatel zvolením položky *Tikety* všechny tikety zobrazit na displej (pro testovací účely byl vsazen ještě jeden tiket, viz obr. 2.3j). Po opětovném zobrazení konta uživatele se už zobrazí částka po odečtení sázek (obr. 2.3k). Vraťme se k zobrazení všech tiketů. Pokud se uživatel rozhodne podívat, které zápasy jsou na některém z tiketů, do pole *ID tiketu pro detail* zadá potřebný údaj a zvolí položku *Detail tiketu.* Po zadání čísla tiketu, které se neshoduje s ani jedním číslem tiketů, které uživatel má, je zobrazena chybová hláška, v opačném případě je zobrazen detail požadovaného tiketu (obr. 2.3m).



Obr. 2.3j

Obr. 2.3k

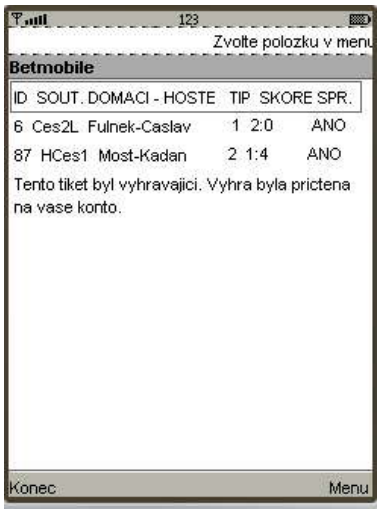


Obr. 2.3m

2.3.4 Kontrola tiketu, kontrola stavu konta, odhlášení

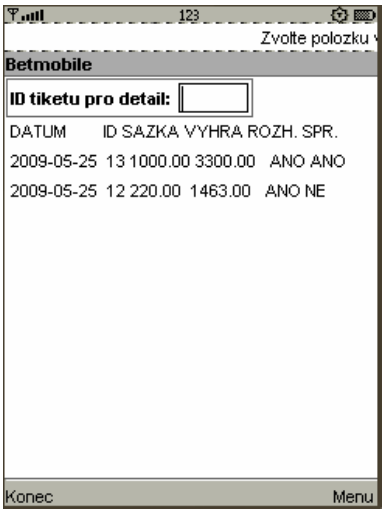
Kontrola tiketu probíhá pokaždé, když se uživatel podívá na detail tiketu. Skript, který kontrolu provádí, porovnává stavy zápasů v nabídce se stavy zápasů na uživatelově tiketu. Pokud tedy administrátor sázkové společnosti zadá do databáze výsledek zápasu, změní se i stav zápasu v nabídce na stav indikující dohranost zápasu. Výsledek se pak porovnává s tipem na uživatelově tiketu. Pokud jsou všechny zápasy na tiketu zkontrolovány, změní se u tiketu sloupec *ROZH*. na hodnotu *ANO* (obr. 2.3p). Do databáze byly tedy výsledky zápasů zadány tak, aby byl tiket číslo *13* výherní (obr. 2.3n) a tiket číslo *14* prohrávající (obr. 2.3o). Uživatel byl o těchto skutečnostech ihned informován na displeji. V prvním případě se tedy vyhraná částka připsala na konto uživatele (obr. 2.3q). Pokud se uživatel podívá na zkontrolovaný tiket (respektive na jeho detail) znova, je informován hláškou, že tiket už viděl (obr. 2.3r). Poslední položka v menu, která nebyla vysvětlena, je položka *Odhlasit.* Jak již bylo předesláno, po zvolení této položky se vymažou data z tabulky BETMOBILE_PRIHLASENI a

ROZPRACOVANE_TIKETY. Tím činnost uživatele končí a je mu zobrazen znova přihlašovací formulář.



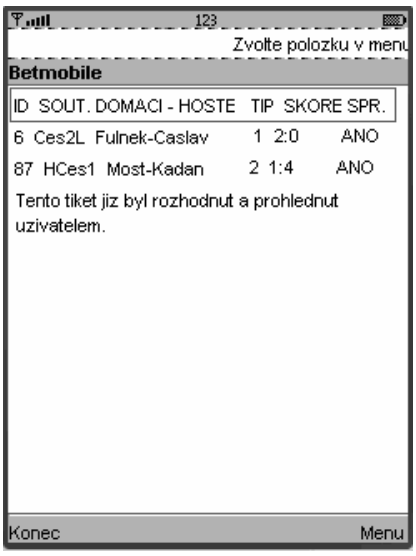
Obr. 2.3n

Obr. 2.3o



Obr. 2.3p

Obr. 2.3q



Obr. 2.3r

3 ZÁVĚR

Cílem této bakalářské práce bylo vyvinout libovolným způsobem aplikaci pro mobilní terminál, která by umožňovala uživateli sázet na sportovní zápasy. Byl zvolen způsob, kdy midlet, neboli aplikace běžící v mobilním terminálu, otevírá HTTP připojení na dle potřeby sestavenou URL adresu. V této adrese je obsaženo umístění skriptu, který se stará o veškeré výpočty a operace s databází. Aby skript věděl, kterou operaci si uživatel skrz midlet vyžádal, načte si (opět z adresy URL) potřebné parametry. Po dokončení požadované operace vrátí skript výslednou informaci zpět mobilnímu terminálu. Ten ji zpracuje a zobrazí uživateli.

Vývoj aplikace byl téměř bezproblémový, až na správné formátování výsledků na displej emulátoru. To proto, že neexistuje nástroj, kterým by se daly měnit atributy písma (např. nastavení konstantní šířky). Až na tuto skutečnost vše fungovalo podle toho, jak to bylo před realizací naplánováno. Nejobtížnější z celého projektu bylo vymyslet systém kontroly tiketů.

Během testování byly stejné výsledky dosaženy jak při použití serveru *localhost* přímo ve zdrojovém počítači, tak při testování na vytvořené WWW stránce, kam byl skript nahrán. Na této stránce bylo pomocí jazyka PHP vytvořeno jednoduché prostředí, které umožňuje registraci nových uživatelů, jejich přihlašování a také sázení. Je zde také umožněna administrace zápasů, kde může administrátor přidávat nové zápasy do databáze, přidávat nové soutěže, editovat zápasy nebo je odstraňovat a hlavně zadávat jejich výsledky. Registrovaný uživatel se tak ke svému účtu může připojit jak přes tuto stránku, tak přes emulátor ve vývojovém prostředí NetBeans. Ukázka WWW stránky je obsažena v příloze (velikost obrázků je upravena).

Seznam použitých zkratek

API Application Programming Interface

BCNF Boyce-Coddova Normální Forma

CDC Connected Device Configuration

CLDC Connected Limited Device Configuration

DDL Data Definition Language

DML Data Manipulation Language

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

J2EE Java 2 Enterprise Edition

J2ME Java 2 Micro Edition

J2SE Java 2 Standard Edition

JDK Java Development Kit

JIT Just In Time

JSP Java Server Pages

JVM Java Virtual Machine

MIDP Mobile Information Device Profile

NF Normální Forma

PDA Personal Digital Assistant

PHP Personal Home Page, později Hypertext Preprocessor

RMI Remote Method Invocation

SDL Storage Definition Language

SQL Structured Query Language

SŘBD Systém Řízení Báze Dat

STB Set-Top Box

URL Uniform Resource Locator

VDL View Definition Language

WWW World Wide Web

Použitá literatura

[1] TOPLEY, K. *J2ME v kostce*. Praha: Grada Publishing, a.s., 2004. 536 s. ISBN 80-247-0426-9

[2] HEROUT, P. *Učebnice jazyka Java.* České Budějovice: Kopp, 2001. 349 s. ISBN 80-7232- 115-3

[3] GILMORE, W.J. *Velká kniha PHP&MySQL*. Brno: Zoner Press, 2005. 864 s. ISBN 80-86815-53-6

[4] SKŘIVAN, J. *Databáze a jazyk SQL* [online]. 2000, [cit. 2009-05-23]. Dostupný z WWW: http://interval.cz/clanky/databaze-a-jazyk-sql/.

[5] WIKIPEDIE. *Databáze* [online]. 2008, [cit. 2009-05-23]. Dostupný z URL: http://cs.wikipedia.org/wiki/Datab%C3%A1ze .

[6] THE PHP GROUP. *Function Reference* [online]. 2009, [cit. 2009-05-23]. Dostupný z URL: http://cz.php.net/manual/en/funcref.php .

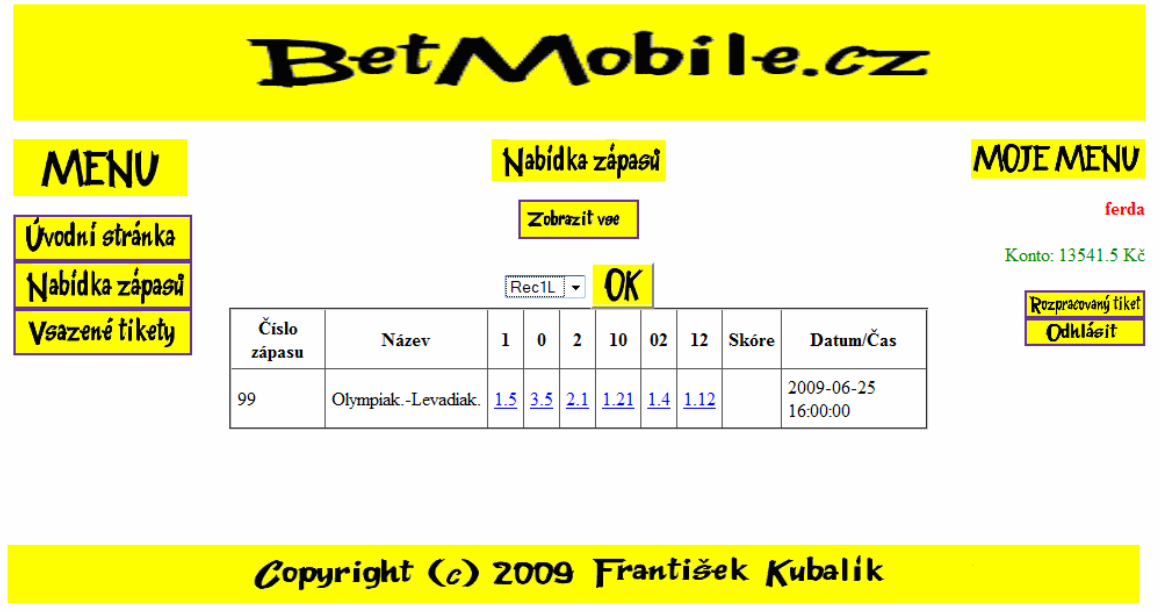
[7] SUN MICROSYSTEMS, INC. *MySQL 5.0 Reference Manual* [online]. 2009, [cit. 2009-05-23]. Dostupný z URL: http://dev.mysql.com/doc/refman/5.0/en/functions.html

[8] BITTNEROVÁ, L.R. *J2ME seriály.* [online]. [cit. 2009-05-23]. Dostupný z URL: http://interval.cz/vyvoj-aplikaci/j2me/

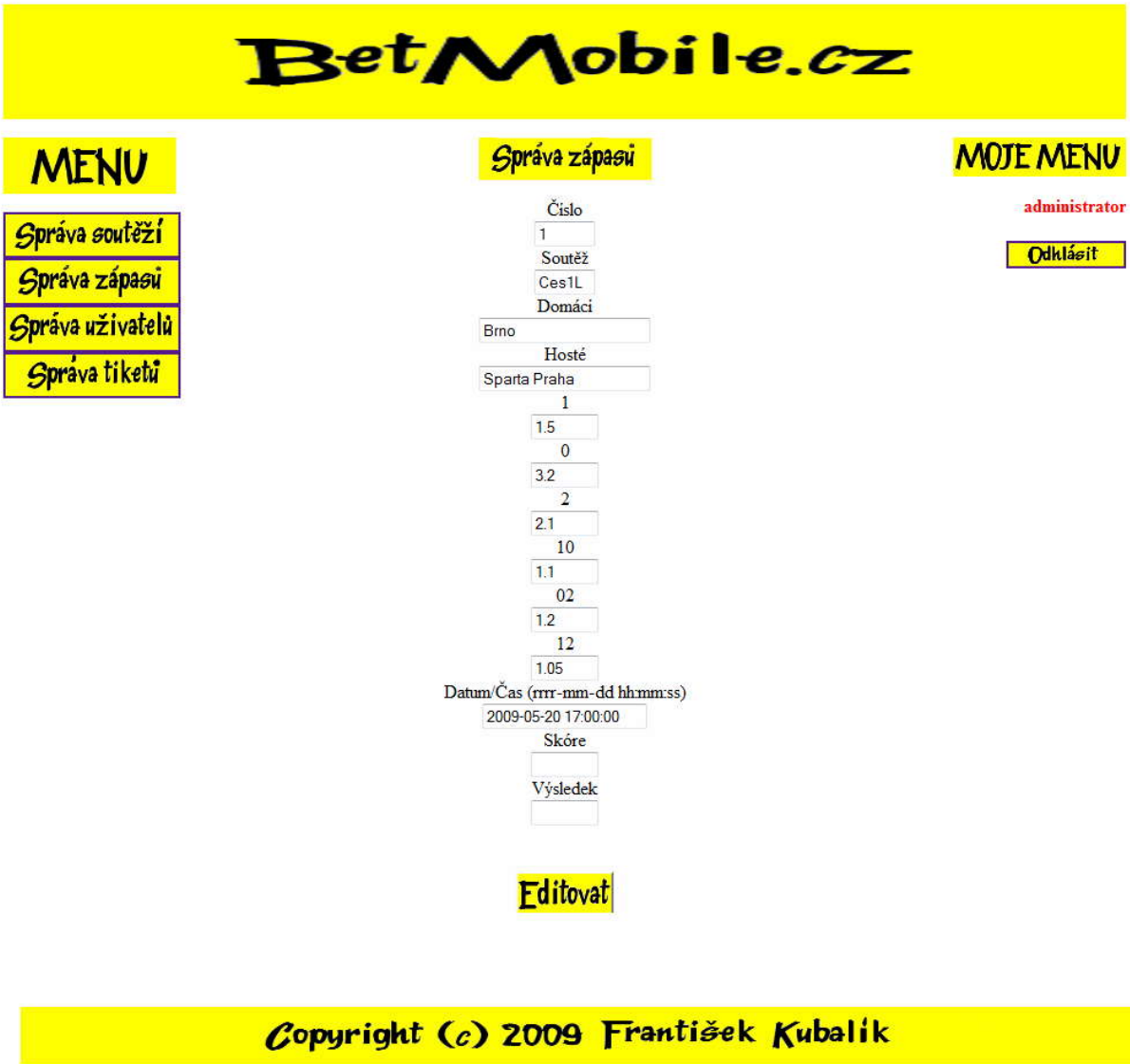
[9] *NetBeans IDE Java Quick Start Tutorial* . [online]. Dostupný z URL: http://netbeans.org

Příloha


BetMobile.cz
MENU
Úvodní stránka
Nabídka zápasů
Úvodní stránka
Seznamte se s možnostmi on-line sázení na sportovní události. Navrhnete aplikaci pro mobilní terminály, která bude umožnovat uživateli sázet na aktuální sportovní zápasy. Daný návrh realizujte v jazyku JavaME. Funkcnost aplikace bude závislá na centrální databázi, kterou taktéž vytvoríte. Bude obsahovat prihlašovací údaje jednotlivých klientu, výši kreditu a kurzy na aktuální sportovní zápasy. Seznamte se s možnostmi on-line sázení na sportovní události. Navrhnete aplikaci pro mobilní terminály, která bude umožnovat uživateli sázet na aktuální sportovní zápasy. Daný návrh realizujte v jazyku JavaME. Funkcnost aplikace bude závislá na centrální databázi, kterou taktéž vytvoríte. Bude obsahovat prihlašovací údaje jednotlivých klientu, výši kreditu a kurzy na aktuální sportovní zápasy. Seznamte se s možnostmi on-line sázení na sportovní události. Navrhnete aplikaci pro mobilní terminály, která bude umožnovat uživateli sázet na aktuální sportovní zápasy. Daný návrh realizujte v jazyku JavaME. Funkcnost aplikace bude závislá na centrální databázi, kterou taktéž vytvoríte. Bude obsahovat prihlašovací údaje jednotlivých klientu, výši kreditu a kurzy na aktuální sportovní zápasy. Seznamte se s možnostmi on-line sázení na sportovní události. Navrhnete aplikaci pro mobilní terminály, která bude umožnovat uživateli sázet na aktuální sportovní zápasy. Daný návrh realizujte v jazyku JavaME. Funkcnost aplikace bude závislá na centrální databázi, kterou taktéž vytvoríte. Bude obsahovat prihlašovací údaje jednotlivých klientu, výši kreditu a kurzy na aktuální sportovní zápasy.
Přihlášení
Login
Heslo
Registrace
OK
Copyright (c) 2009 František Kubalík


BetMobile.cz
MENU
Úvodní stránka
Nabídka zápasů
Vsazené tikety
Nabídka zápasů
Zobrazit vse
Rec1L
OK
MOJE MENU
ferda
Konto: 13541.5 Kč
Rozpracovaný tiket
Odhlásit
Číslo zápasu | Název | 1 | 0 | 2 | 10 | 02 | 12 | Skóre | Datum/Čas
99 | Olympiak.-Levadiak. | 1.5 | 3.5 | 2.1 | 1.21 | 1.4 | 1.12 | | 2009-06-25 16:00:00
Copyright (c) 2009 František Kubalík


BetMobile.cz
MENU
Úvodní stránka
Nabídka zápasů
Vsazené tikety
Vsazené tikety
Datum/Čas
Sázka
Max.kurz
Výhra
Rozhodnuto
Správnost
2009-05-25 18:51:50
1000.00
3.30
3300.00
ANO
ANO
0.
2009-05-25 18:44:29
220.00
6.65
1463.00
ANO
NE
0.
MOJE MENU
ferda
Konto: 13541.5 Kč
Rozpracovaný tiket
Odhlásit
Copyright (c) 2009 František Kubalík


BetMobile.cz
MENU
Správa soutěží
Správa zápasů
Správa uživatelů
Správa tiketů
Správa zápasu
Číslo
1
Soutěž
Ces1L
Domáci
Brno
Hosté
Sparta Praha
1
1.5
0
3.2
2
2.1
10
1.1
02
1.2
12
1.05
Datum/Čas (rrrr-mm-dd hh:mm:ss)
2009-05-20 17:00:00
Skóre
Výsledek
Editovat
MOJE MENU
administrator
Odhlásit
Copyright (c) 2009 František Kubalík